(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 628 455**

(51) Int. CI.:

G06F 9/50 (2006.01)
G06F 17/30 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **17.06.2011 PCT/EP2011/003011**

(87) Fecha y número de publicación internacional: **22.12.2011 WO11157442**

(96) Fecha de presentación y número de la solicitud europea: **17.06.2011 E 11729557 (6)**

(97) Fecha y número de publicación de la concesión europea: **12.04.2017 EP 2583175**

(54) Título: **Procesamiento paralelo de consultas continuas sobre flujos de datos**

(30) Prioridad:

20.05.2011 US 201113112628
18.06.2010 US 356353 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
02.08.2017

(73) Titular/es:

UNIVERSIDAD POLITÉCNICA DE MADRID (100.0%)
C/ Ramirez de Maeztu n°7
28040 Madrid, ES

(72) Inventor/es:

JIMENEZ PERIS, RICARDO y
PATIÑO MARTINEZ, MARTA

(74) Agente/Representante:

ARIAS SANZ, Juan

DESCRIPCIÓN

Procesamiento paralelo de consultas continuas sobre flujos de datos

Campo de la invención

La presente invención pertenece a los campos del procesamiento de flujos de datos y de la gestión de eventos.

Estado de la técnica

Los motores de procesamiento de consultas continuas permiten el procesamiento de flujos de datos mediante consultas que procesan de forma continua esos flujos, produciendo unos resultados que se actualizan con la llegada de nuevos datos en el flujo de datos. Motores de procesamiento de consultas continuas conocidos son Borealis (Daniel J. Abadi, Yanif Ahmad, Magdalena Balazinska, Ugur Çetintemel, Mitch Cherniack, Jeong-Hyon Hwang, Wolfgang Lindner, Anurag Maskey, Alex Rasin, Esther Ryvkina, Nesime Tatbul, Ying Xing, Stanley B. Zdonik: *The Design of the Borealis Stream Processing Engine*. CIDR 2005: 277 - 289), Aurora (Daniel J. Abadi, Donald Carney, Ugur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, Stanley B. Zdonik: *Aurora: a new model and architecture for data stream management*. VLDB J. 12 (2): 120 - 139 (2003)) y TelegraphCQ (Sirish Chandrasekaran, Owen Cooper, Amol Deshpande, Michael J. Franklin, Joseph M. Hellerstein, Wei Hong, Sailesh Krishnamurthy, Samuel Madden, Vijayshankar Raman, Frederick Reiss, Mehul A. Shah: *TelegraphCQ: Continuous Dataflow Processing for an Uncertain World*. CIDR 2003). A diferencia de las bases de datos que realizan consultas instantáneas sobre datos persistentes, los motores de procesamiento de flujos realizan consultas continuas sobre flujos de datos que evolucionan con el tiempo y se procesan en memoria sin almacenarse en disco, y sus resultados también evolucionan con el tiempo.

Ninguno de los enfoques existentes en la actualidad permite escalar con respecto al volumen de los flujos de entrada. Esto es debido a que el flujo de datos procesado por una consulta o una consulta de operador debe pasar por un único nodo, que contiene la consulta u operador y, por lo tanto, la capacidad del sistema estará limitada por la capacidad de un único nodo. Para volúmenes de flujo que superen la capacidad de procesamiento de un nodo, estos sistemas no pueden escalar. En motores centralizados tales como Aurora o TelegraphCQ, la totalidad del flujo pasa por el nodo centralizado del sistema y, cuando este nodo se satura, no puede escalar. En motores distribuidos como Borealis, un operador de consulta se despliega en un único nodo. Esto implica que la totalidad del flujo que es procesado por ese operador es procesado por el nodo en el que se ejecuta el mismo y, por lo tanto, cuando se satura el nodo, el sistema no puede escalar. En algunos sistemas se está tratando de introducir un cierto paralelismo en consultas tales como Aurora * (Mitch Cherniack, Han Balakrishnan, Magdalena Balazinska, Donald Camey, Ugur Çetintemel, Ying Xing, Stanley B. Zdonik: *Scalable Distributed Stream Processing*. CIDR 2003). En Aurora*, se emplea un nodo para distribuir la carga entre un conjunto de nodos y otro nodo para recombinar la misma. Estos dos nodos se vuelven, una vez más, un cuello de botella debido a que, una vez que se han saturado, el sistema no puede escalar.

El equilibrado de carga también se ha estudiado en el contexto de los motores de procesamiento de flujo. En Ying Xing, Stanley B. Zdonik, Jeong-Hyon Hwang: *Dynamic Load Distribution in the Borealis Stream Processor*. ICDE 2005: 791 - 802, se estudia la correlación entre las cargas de trabajo para evitar picos de carga de trabajo en diferentes operadores que tienen lugar en el mismo nodo. No obstante, este equilibrado de carga se estudia en el contexto de un motor de consultas distribuido que no paraleliza las consultas, por lo tanto este no aborda el problema de cómo distribuir la carga entre instancias de la misma subconsulta, sino entre subconsultas diferentes.

Otra técnica que se emplea en la actualidad para tratar las sobrecargas es la eliminación de ítems de datos o tuplas, que se conoce como compartimiento de carga (*load shedding)* (Frederick Reiss, Joseph M. Hellerstein: *Data Triage: An Adaptive Architecture for Load Shedding in TelegraphCQ*. ICDE 2005: 155 - 156), (Frederick Reiss, Joseph M. Hellerstein: *Data Triage: An Adaptive Architecture for Load Shedding in TelegraphCQ*. ICDE 2005: 155 - 156) y (Nesime Tatbul, Ugur Çetintemel, Stanley B. Zdonik: *Staying FIT: Efficient Load Shedding Techniques for Distributed Stream Processing*. VLDB 2007: 159 - 170). Con esta técnica, cuando un nodo se satura, este comienza a descartar tuplas de acuerdo con diferentes criterios. El problema con esta técnica es que la pérdida de información que no es admisible para una multitud de aplicaciones y tiene además contrapartidas asociadas tales como la pérdida de precisión en el resultado de las consultas o incluso la pérdida de coherencia en el resultado de las consultas.

Otra técnica propuesta recientemente es el empleo de aceleración de hardware mediante la implementación de los operadores de flujos de datos en hardware con FPGA (René Müller, Jens Teubner, Gustavo Alonso: *Streams on Wires - A Query Compiler for FPGAs*. PVLDB 2 (1): 229 - 240 (2009). Este enfoque de hardware permite mejorar el rendimiento de nodos individuales pero no escalar. Es decir, este hace que un nodo tenga una capacidad superior pero una vez que se ha consumido esa capacidad, este enfoque impide escalar más allá.

La presente invención se contempla como una necesidad con respecto a los métodos y motores de procesamiento existentes en el mercado. La invención paraleliza el procesamiento de consultas continuas de una forma escalable, con un coste de distribución bajo e introduce elasticidad y equilibrado de carga en el procesamiento paralelo de

consultas.

Descripción de la invención

La invención presenta un motor paralelo de consultas continuas sobre flujos de datos. Este motor de procesamiento paralelo resolverá las limitaciones del estado de la técnica, evitando que el motor de consultas de flujos de datos procese el flujo completo con nodos individuales, que limita la capacidad del sistema a la capacidad de procesamiento de un único nodo.

El procesamiento paralelo de flujos de datos permite proporcionar escalabilidad y, de esa forma, aumentar el rendimiento por medio de la adición de nuevos nodos. Este procesamiento paralelo se puede aplicar al procesamiento de flujos de datos y al procesamiento de eventos complejos.

La invención se expone en la reivindicación independiente de método 7 y en la reivindicación independiente de sistema 1 que se refieren a un motor de procesamiento paralelo de flujos.

Un flujo de datos es una secuencia de ítems de datos o tuplas, que con el tiempo, puede crecer de forma ilimitada y que se caracteriza por la compartición del mismo esquema de datos por parte de todos los ítems de datos en el mismo flujo. En muchos motores de procesamiento de flujos de datos, los flujos de datos se marcan con una marca de tiempo. Dependiendo de la aplicación, el flujo puede garantizar que los datos se marcan con marcas monótonamente crecientes.

Un motor de procesamiento de flujos de datos o sistema de gestión de flujos de datos es un sistema que permite desplegar consultas continuas sobre flujos de datos. El motor de procesamiento etiqueta las tuplas con marcas de tiempo. Estas marcas de tiempo permiten establecer un orden relativo entre las tuplas.

Una consulta se representa con un gráfico acíclico de operadores de consulta. La consulta tiene uno o más flujos de entrada de datos y uno o más flujos de salida de datos. Cada operador de consulta puede tener una o más entradas y una o más salidas. Los operadores se pueden clasificarse como operadores sin estado u operadores con estado.

Los operadores sin estado se caracterizan por que para cada tupla de entrada produce una salida sin depender de ninguna otra tupla de entrada previa. Operadores de consulta típicos sin estado son transformación (*map*), filtrado (*filter*) y unión (*union*). Un operador de transformación aplica una función a cada tupla de entrada con el fin de obtener la correspondiente tupla de salida. Por ejemplo, dada una tupla con una temperatura en grados Farenheit este puede producir una tupla con la temperatura en grados centígrados. El operador de filtrado, dada una secuencia de n predicados y n o n + 1 salidas, ejecuta el siguiente procesamiento con cada tupla. Este aplica el primer predicado, si el predicado se cumple emite esa tupla a través de la primera salida. Si no, aplica el segundo predicado y, si este se cumple, emite esa tupla a través de la segunda salida. Y así sucesivamente, con el resto de los predicados. Si el número de predicados es igual al número de salidas y no se cumple predicado alguno, la tupla de entrada se descarta. Si el número de salidas es n + 1 y no se cumple ningún predicado, la tupla se emite a través de la salida n + 1. La unión, dados dos o más flujos de entrada con el mismo esquema, produce un único flujo de salida con ese esquema y en el cual se emiten todas las tuplas recibidas por los diferentes flujos de entrada.

Los operadores con estado mantienen una ventana deslizante de tuplas de entrada y el resultado del procesamiento de una tupla de entrada no solo depende de esta sino del contenido de la ventana de tuplas. Las ventanas deslizantes se pueden definir sobre el tiempo o sobre el número de tuplas. El lapso de tiempo o el número de tuplas que admite la ventana se conoce como longitud de la ventana. Algunos operadores con estado típicos son el operador de agregación (*aggregation*), y el operador de reunión (*join*). El operador de agregación computa una función sobre las tuplas contenidas en la ventana de entrada, por ejemplo, el número de tuplas recibidas en la última hora, o la temperatura promedio con respecto a la temperatura indicada por las tuplas recibidas durante las últimas 24 horas. El operador de reunión recibe dos flujos de entrada y mantiene una ventana para cada flujo. El operador tiene como parámetro un predicado. Para cada par de tuplas, una de cada ventana de entrada, este aplica el predicado y, si el par de tuplas lo cumple, genera una tupla de salida que es la concatenación de las dos tuplas de entrada. Si las ventanas deslizantes de entrada de la entrada del operador de reunión se definen en función del tiempo, estas evolucionan del siguiente modo, cuando llega una tupla en un flujo de entrada, se eliminan de la otra ventana todas las tuplas con una marca de tiempo más grande que la longitud temporal de la ventana.

Los motores de procesamiento de flujos pueden ser centralizados o distribuidos. Un motor de procesamiento de flujos centralizado tiene una sola instancia de sistema, que se ejecuta en un único ordenador o nodo. Es decir, el sistema se ejecuta en un único nodo. Un motor de procesamiento de flujos distribuido tiene múltiples instancias, es decir, se realizan múltiples ejecuciones del sistema y cada instancia puede ser ejecutada por diferentes nodos. Los motores distribuidos más básicos pueden ejecutar diferentes consultas en diferentes nodos. De este modo, pueden escalar el número de consultas mediante el aumento del número de nodos. Algunos motores distribuidos permiten distribuir los operadores de consulta en diferentes nodos. Esto permite que los mismos escalen con respecto al número de operadores mediante el aumento del número de nodos.

No obstante, la presente invención va más allá del estado de la técnica al introducir un motor de procesamiento de consultas continuas distribuido en paralelo. Esto quiere decir que, por un lado, se ejecutan múltiples instancias del motor de procesamiento en múltiples nodos. Por otro lado, múltiples instancias del motor cooperan para procesar una subconsulta repartiendo el flujo de entrada y, de este modo, escalando con respecto al volumen del flujo de entrada.

Una de las principales dificultades a solucionar y una de las principales contribuciones de la presente invención es cómo procesar en paralelo uno o más flujos masivos de datos por parte de un conjunto de nodos sin concentrar ninguno de los flujos en nodo individual alguno.

Una consulta continua es, de forma abstracta, un grafo acíclico de operadores de consulta. Esta consulta se puede dividir en un conjunto de subconsultas al dividir el grafo de consulta en un conjunto de subgrafos.

Una vez subdividida en un conjunto de subconsultas, la consulta se despliega. Esta división se puede realizar de acuerdo con cualquier criterio. Algunos criterios posibles son:

1) no dividir la consulta de origen, es decir, el resultado de la división sería una subconsulta idéntica a la consulta de origen original completa;
2) dividir la consulta de origen en tantas subconsultas como operadores de consulta comprende la misma, consistiendo cada subconsulta en cada uno de los operadores de consulta que aparecen en la consulta de origen;
3) dividir la consulta de origen en subconsultas, de tal modo que cada subconsulta contenga un operador con estado, seguido por uno o más operadores sin estado, posiblemente excepto por una subconsulta inicial que contenga solo operadores sin estado;
4) cualquier otra subdivisión en subconsultas.

El procedimiento de procesamiento paralelo que implementa el motor de procesamiento paralelo de flujos despliega cada subconsulta en un conjunto de nodos, de tal modo que cada subconsulta se ejecuta en al menos un nodo. Cada conjunto de nodos puede tener un número arbitrario de nodos y puede tener un número diferente de nodos en cada conjunto de nodos, y cambiar de forma dinámica su número de forma independiente.

Si una consulta se divide en dos o más subconsultas, cada par de subconsultas consecutivas, subconsulta de origen y subconsulta de destino, en el que las salidas de la subconsulta de origen están conectadas con una o más entradas de la subconsulta de destino, estas se conectarán en su despliegue distribuido en paralelo tal como sigue. Cada una de las subconsultas, debido a que la misma se despliega en un conjunto de nodos, la conexión se hará desde cada instancia de la subconsulta de origen a cada instancia de la subconsulta de destino.

El procedimiento de procesamiento paralelo considera dos métodos de procesamiento de consultas dependiendo de cómo se subdivide la consulta de origen. Si todas las subconsultas cumplen la condición de contener como mucho un operador con estado y su entrada o entradas provienen de subconsultas previas, entonces el procesamiento paralelo redistribuye los flujos entre subconsultas en el origen y, en caso contrario, en el destino.

El procesamiento paralelo con la redistribución en origen consiste en interponer un equilibrador de carga a la salida de cada instancia de la subconsulta de origen, que se denomina subconsulta local al equilibrador de carga. Es decir, en cada nodo de la subconsulta de origen se interpone un equilibrador de carga de tal modo que, en cada instancia, la salida de la subconsulta de origen se conecta a la entrada del equilibrador de carga y su salida se conecta a la totalidad de las instancias de la subconsulta de destino. En cada instancia de la subconsulta de destino se interpone un mezclador de entradas, de tal modo que la totalidad de las salidas de la subconsulta de origen pasan a ser entradas del mezclador de entradas y su salida pasa a ser la entrada en la instancia de la subconsulta de destino. La instancia de la consulta que está conectada con el mezclador de entradas se denomina instancia local.

El procesamiento paralelo con redistribución en destino, además de interponer un par de equilibrador de carga y de mezclador de entradas entre subconsultas, estos también están interpuestos en cada subconsulta antes de cada operador con estado que esté precedido por cualquier otro operador en la subconsulta. Esto quiere decir que una tupla con redistribución en destino puede tener que pasar por múltiples instancias hasta completar su procesamiento por parte de la subconsulta de destino.

Dada una subconsulta desplegada en un conjunto de nodos que cumpla la condición de redistribución en origen, el procesamiento entre sus diferentes instancias se realiza del siguiente modo. El primer operador de la subconsulta de destino es un operador con estado. El operador con estado ejecuta su operación en función de uno o más campos clave. Las tuplas con la misma clave se han de agregar (la agregación es solo un ejemplo debido a que esta podría ser cualquier operación ejecutada por un operador con estado) de forma conjunta. Esto quiere decir que estas han de ser recibidas por la misma instancia de tal modo que las mismas se puedan agregar. La redistribución consciente de la semántica distribuye las tuplas de tal modo que las tuplas con la misma clave son recibidas por la misma instancia del operador de consulta con estado. Si la estrategia empleada por los equilibradores de carga da como resultado una redistribución consciente de la semántica, se dice que el equilibrador de carga es consciente de la

semántica.

En la estrategia de paralelización que se describe, los equilibradores de carga que envían tuplas a una subconsulta de destino que comienza con un operador de consulta con estado son conscientes de la semántica. Se puede emplear cualquier método de distribución de la carga consciente de la semántica. Por simplicidad, en la descripción, cuando haga falta hacer referencia a un método de distribución de la carga, se asumirá el método de distribución de la carga que se detalla a continuación. Dado un flujo con una clave c, un posible método de redistribución es en el que, en cada tupla con clave c, se aplica una función hash para obtener un valor h = hash (c). A partir de este valor h, se aplica la operación resto con una constante np, obteniendo un valor p = h mod np. Este valor p se denominará identificador de partición. Dada una tupla, aplicando el procedimiento anterior, se obtiene su identificador de partición. El número total de particiones es np, y cada partición con el identificador p se asigna a una instancia de subconsulta. La responsabilidad de procesar las tuplas procedentes de una partición corresponde, por lo tanto, a una única instancia. De esta forma, el método de distribución cumple el requisito de consciencia de la semántica debido a que todas las tuplas que se han de agregar de forma conjunta se enviarán a la misma instancia.

Con independencia del tipo de redistribución, la paralelización se puede realizar con o sin garantías de una paralelización transparente. En el segundo caso, la paralelización se realiza para permitir la consciencia de la semántica de los operadores con estado, pero esta no garantiza que la ejecución paralela resultante sea equivalente a una ejecución en un motor secuencial. En el primer caso, esta garantiza que el procesamiento paralelo sea equivalente al que se observaría en un motor secuencial. Este aspecto de la paralelización se ve reflejado en el equilibrador de carga y en el mezclador de entradas.

Cuando no hay garantía alguna de paralelización transparente, el mezclador de entradas reenvía las tuplas tan pronto como estas se reciben de cualquiera de las subconsultas de origen a la subconsulta de destino a la que está conectada o instancia local de la subconsulta. Esto puede producir una intercalación que nunca se produciría en un motor secuencial. La razón de que estas intercalaciones se puedan producir se debe a que las ventanas de cada instancia de un operador con estado paralelo, por ejemplo, un operador de reunión, deslizan de forma independiente. Es decir, una intercalación (orden relativo) de las tuplas en los dos flujos de entrada en el caso secuencial produciría una secuencia de ventanas solapantes con la correspondiente salida, en el caso paralelo, la intercalación de las tuplas en los dos flujos de entrada puede ser diferente en cada instancia, produciendo diferentes secuencias de ventanas solapantes y, por lo tanto, esta podría producir diferentes salidas.

Para garantizar la transparencia de la paralelización, el equilibrador de carga y el mezclador de entradas funcionan de la forma que sigue. Cada mezclador de entradas espera hasta haber recibido una tupla de cada flujo de entrada antes de reenviar una tupla a la subconsulta a la que está conectada o subconsulta local. Este deja pasar la tupla con la marca de tiempo más pequeña. De esta forma, el procesamiento paralelo en cada instancia pasa a ser independiente de la intercalación de los flujos procedentes de los diferentes equilibradores de carga. Este procedimiento puede dar lugar a bloqueo.

Si alguna subconsulta de origen no produce tuplas a ser procesadas por la subconsulta de destino, entonces el mezclador de entradas se bloqueará. Para evitar esta situación, los equilibradores de carga funcionarían del siguiente modo. Cada equilibrador de carga realiza un seguimiento de la marca de tiempo de la última tupla generada para cada subconsulta de destino. Cuando a una subconsulta de destino no se le ha enviado tupla alguna durante un periodo de tiempo máximo m, entonces este envía una tupla espuria con una marca de tiempo idéntica a la última enviada por ese equilibrador de carga. Cuando la tupla espuria es recibida por un mezclador de entradas, esta se emplea solo para desbloquear el procesamiento del mezclador de entradas. Si esta no tiene la marca de tiempo más pequeña, el mezclador de entradas cogerá la tupla que tenga la marca de tiempo más pequeña. Tarde o temprano, la tupla espuria pasará a ser la que tenga la marca de tiempo más pequeña, en ese caso el mezclador de entradas simplemente la descartará. De esta forma, la generación periódica de tuplas espurias en los equilibradores de carga evita bloquear el mezclador de entradas.

La elasticidad es una propiedad de los sistemas distribuidos que hace referencia a la capacidad de aumentar y disminuir el número de nodos para procesar la carga entrante mediante el empleo de los mínimos recursos necesarios, es decir, el mínimo número posible de nodos capaz de procesar la carga entrante cumpliendo los requisitos de calidad de servicio.

El equilibrado de carga en un sistema distribuido hace referencia al método que se emplea para distribuir la carga a procesar por los diferentes nodos de tal modo que todos los nodos tengan una carga similar. Cuando los nodos tienen una capacidad de procesamiento diferente, el objetivo es equilibrar es la carga relativa, es decir, que cada nodo emplee la misma fracción de su capacidad de procesamiento. El equilibrado de carga puede ser estático o dinámico. El equilibrado de carga estático se decide antes de desplegar el sistema y este no varía durante la ejecución. El equilibrado de carga dinámico está variando de forma continua durante el tiempo de ejecución, lo que permite la adaptación a los cambios en la carga de trabajo. Una propiedad muy importante del equilibrado de carga dinámico es que este ha de afectar lo menos posible a la capacidad de procesamiento de los nodos.

La elasticidad y el equilibrado de carga están íntimamente relacionados entre sí debido a que ambas propiedades

necesitan de una técnica común que se conoce como transferencia de estado. La transferencia de estado consiste es transferir parte o la totalidad de los datos de un nodo a otro. Una vez que la transferencia de estado a un nodo se ha completado, ese nodo es responsable de procesar la carga para ese estado. En el contexto de la presente invención, el estado de un operador de consulta con estado consiste en la ventana deslizante de tuplas y cualquier valor necesario para generar la salida, por ejemplo, el valor actual agregado en un operador de agregación (por ejemplo, las temperaturas promedio durante la última hora).

El procedimiento de transferencia de estado consiste en las siguientes etapas. Siempre que se reconfigura el motor de procesamiento por el motivo que sea (por ejemplo: desequilibrio en la carga, fallo de un nodo, se añade un nuevo nodo para evitar la sobrecarga), el procedimiento de reconfiguración reasigna particiones de datos de una o más instancias a una o más instancias. Este problema se puede descomponer en transferencias de particiones de datos individuales de una instancia de origen a una instancia de destino. Esto quiere decir que el estado de esa partición de datos se ha de transferir de la instancia de origen a la instancia de destino. Para ello se examina la marca de tiempo activa más grande en el sistema, mt, y se crea una nueva marca de tiempo futura, mtf = mt + f, en la que $f \in N$ que proporciona un margen suficiente para alertar a todas las instancias involucradas en la reconfiguración acerca de la marca de tiempo que inicia la reconfiguración. Esta marca de tiempo se envía a todas las instancias involucradas, es decir, a todas las instancias de la subconsulta de origen, así como a las dos instancias de la subconsulta de destino que realizan la transferencia de estado. Todas las tuplas con una marca de tiempo más pequeña o igual que mtf son procesadas por la instancia de origen, mientras que todas las tuplas con una marca de tiempo más grande que mtf son procesadas por las instancias de destino.

Los equilibradores de carga de la subconsulta de origen almacenan la marca de tiempo mtf incluida en la orden de reconfiguración. Las tuplas que pertenecen a la partición que se está reconfigurando, p, con una marca de tiempo más pequeña o igual que mtf se envían a la instancia de la subconsulta de destino responsable de la partición p antes de la reconfiguración, mientras que las tuplas con una marca de tiempo más grande que mtf se envían a la instancia de la subconsulta de destino responsable de la partición p después de la reconfiguración. Cuando se va a enviar la primera tupla de la partición p más grande que mtf, en primer lugar se envía una tupla que indica que el fin de la reconfiguración a las dos instancias de la subconsulta de destino involucradas en la transferencia de estado.

La transferencia de estado se completa con las siguientes etapas. La instancia de origen de la transferencia de estado procesa las tuplas recibidas. Cuando se recibe la tupla de fin de reconfiguración de todos los equilibradores de carga de la subconsulta de origen, tiene la certeza de que ya no debe procesar tuplas adicionales, por lo que transfiere el estado de la partición p a la instancia de destino de la transferencia de estado. La instancia de destino de la transferencia de estado, cuando recibe la transferencia de estado de la partición p, aplica ese estado y almacena la responsabilidad de la partición p. Cuando recibe las tuplas de fin de reconfiguración de todos los equilibradores de carga de la subconsulta de origen, inicia el procesamiento de la partición p. En este momento, la transferencia de estado se ha completado y la responsabilidad de la partición p ha pasado de la instancia de origen a la de destino.

El método para conseguir elasticidad y distribución de carga se describe a continuación. En primer lugar, cada instancia monitoriza localmente su utilización de CPU y de memoria. Para cada subconsulta, uno de los nodos del conjunto de nodos en el que se ha desplegado la subconsulta es responsable de compilar la información de carga de todos los nodos en los que está desplegada la subconsulta. Un procedimiento especial, que se denomina aprovisionador de subconsulta, se encarga de esta tarea en ese nodo. Todos los nodos de una subconsulta envían de forma periódica los datos de monitorización de carga al aprovisionador. Como parte del mensaje de monitorización de carga, un nodo también envía la marca de tiempo más grande de entre las tuplas que ha procesado el mismo. El aprovisionador compara la carga relativa de los diferentes nodos. Si el desequilibrio entre los nodos supera un primer umbral de desequilibrio dado, el proveedor decide cómo reequilibrar las particiones para equilibrar la carga. Una vez que se ha tomado la decisión, este obtiene la marca de tiempo más grande conocida y envía una orden de reconfiguración a los equilibradores de carga de la subconsulta de origen y a las instancias de subconsulta que serán reconfiguradas. La transferencia de estado se realiza de acuerdo con el método que se ha mencionado en lo que antecede.

El aprovisionador también comprueba si la carga promedio de los nodos supera un segundo umbral de utilización máximo. Si esto tiene lugar, quiere decir que el conjunto de nodos que ejecuta la subconsulta está cercano a la saturación y se ha de añadir un nuevo nodo con el fin de añadir capacidad de computación. Entonces, la subconsulta se despliega en un nodo de entre el conjunto de nodos disponibles. Una vez que el nuevo nodo está listo para ejecutar la subconsulta, el aprovisionador incluye el nuevo nodo en el conjunto de nodos que ejecutan la subconsulta. El mecanismo de equilibrado de carga detecta un desequilibrio entre el nuevo nodo y el resto de los nodos e inicia de inmediato la reconfiguración aplicando el método de equilibrado de carga que se ha descrito en lo que antecede.

El aprovisionador también comprueba si la carga promedio en su conjunto podría ser procesada por un número de nodos más pequeño sin superar el segundo umbral máximo promedio de carga. Si esto tiene lugar, el aprovisionador selecciona cualquier nodo y reconfigura el sistema de tal modo que todas las particiones de ese nodo se distribuyan de manera uniforme entre el resto de los nodos que procesan la subconsulta. El aprovisionador inicia la

reconfiguración como en el caso anterior y esta se procesa como una reconfiguración de equilibrado de carga. Cuando finaliza la reconfiguración, ese nodo se devuelve al conjunto de nodos disponibles.

Breve descripción de los dibujos

En lo sucesivo, para facilitar la comprensión de la invención, se describe a modo ilustrativo pero no limitativo una realización de la invención. Esta hace referencia a una serie de figuras.

La **figura 1** muestra una consulta con operadores de transformación (M), de filtrado (F), de Reunión (J) y de Agregación (A).
La **figura 2** muestra un conjunto de subconsultas procedentes de la división de la consulta en la figura 1 dado un criterio de división.
La **figura 3** muestra un conjunto de subconsultas procedentes de la división de la consulta en la figura 1 dado otro criterio de división.
La **figura 4** muestra dos subconsultas consecutivas con redistribución en origen.
La **figura 5** muestra dos subconsultas consecutivas con redistribución en el origen.

Descripción detallada de un modo de realización

La figura 1 muestra una consulta con operadores de transformación (M), filtrado (F), Reunión (J) y Agregación (A). En esta consulta, las tuplas entrantes entran a través del operador de la izquierda. El operador de transformación transforma una tupla con la función de transformación asociada. El operador de filtrado aplica un predicado a la tupla, si este se cumple, entonces la tupla se reenvía al siguiente operador, en caso contrario, es descartada. La salida del operador de filtrado está conectada con las dos entradas del operador de reunión. Es decir, cada tupla producida por el operador de filtrado se envía a cada una de las dos entradas del operador de reunión, realizando una auto-reunión. El operador de reunión aplica un predicado a todos los pares que son mantenidos en las dos ventanas deslizantes (asociadas a los flujos de entrada respectivos). Cada par que cumple el predicado se concatena y se genera como una tupla de salida. El siguiente operador es una agregación. Este agrega las tuplas de acuerdo con una función dada o una condición de agrupamiento. Una tupla se genera de forma periódica con el valor agregado después de cada deslizamiento de ventana. Por último, el último operador filtra estas tuplas empleando un predicado.

Las figuras 2 y 3 muestran la misma consulta que se muestra en la figura 1, particionada en subconsultas de acuerdo con dos criterios diferentes. La partición se indica con líneas verticales de trazo discontinuo. Las subconsultas se corresponden con cada fragmento de la subconsulta original que está delimitada por las líneas. La figura 2 muestra una partición en la que cada subconsulta consiste en un operador. La partición en la figura 3 se realiza en función de operadores con estado. Esto ha conducido a tres subconsultas. La primera está constituida por un prefijo de operadores sin estado, los operadores de transformación y de filtrado. La segunda está constituida por los operadores de reunión y de filtrado. La tercera está constituida por los operadores de agregación y de filtrado. Las subconsultas se caracterizan por empezar con un operador con estado seguido de todos los operadores sin estado hasta el siguiente operador con estado. La única excepción es el prefijo de los operadores sin estado hasta el primer operador con estado, tal como tiene lugar con la primera subconsulta en la figura 3.

La figura 4 muestra dos subconsultas consecutivas de una consulta distribuida en paralelo. Dadas dos subconsultas consecutivas, se dice que la primera es la subconsulta de origen (2) de la segunda y que la segunda es la subconsulta de destino (3) de la primera. La subconsulta de origen (2) consiste, en este caso, en un único operador, el operador de transformación (M). La subconsulta de destino (3) consiste en un único operador, el operador de agregación (A). Cada subconsulta se despliega en un conjunto de nodos. En este caso, cada uno de los conjuntos tiene dos nodos (1). Una instancia de la subconsulta se despliega en cada uno de los nodos (1) de cada conjunto de nodos. Cada instancia se ejecuta en un nodo (1) diferente. Cada instancia de una subconsulta se extiende para volverse distribuida en paralelo con dos operadores de distribución. Al inicio de cada instancia de cada subconsulta se introduce un mezclador de entradas (IM). Al final de cada instancia de cada subconsulta se introduce un equilibrador de carga (LB). El equilibrador de carga (LB) se encarga de distribuir las tuplas de salida de cada instancia de una subconsulta a las instancias de la subconsulta de destino 3. El equilibrador de carga (LB) es consciente de la semántica, por lo que distribuye las tuplas que se han de agregar de forma conjunta a la misma instancia de la subconsulta de destino (3).

La figura 5 muestra las subconsultas de origen (2) y de destino (3) que también se muestran en la figura 4, pero con redistribución en destino. Las tuplas son enviadas por los equilibradores de carga (LB) de la subconsulta de origen (2) a cualquier instancia de la subconsulta de destino (3) sin consciencia de la semántica. El operador con estado de la subconsulta de destino (3), en este caso un operador de agregación (A), redistribuye las tuplas a la instancia adecuada con consciencia de la semántica.

El modo de realización preferente de la invención contempla el caso en el que se proporciona una paralelización transparente. El caso sin transparencia es una simplificación de ese.

En este modo de realización se contemplan los operadores de consulta de flujo de datos más populares. Los operadores de consulta sin estado son transformación, filtrado y unión. Los operadores de consulta con estado son agregación y reunión. En el modo de realización, las tuplas incluyen marcas de tiempo de origen. Las marcas de tiempo son monótonamente crecientes.

La semántica asumida para los operadores sin estado es la siguiente. El operador de transformación aplica una función a cada tupla de entrada y genera una tupla de salida de acuerdo con esa función. El operador de filtrado puede tener asociada una secuencia de uno o más predicados y tantas salidas como predicados, o una salida más que predicados. Es decir, dados n predicados, este tendrá n o n + 1 salidas. Para cada tupla de entrada, los predicados se evalúan uno a uno hasta que se cumpla un predicado o no se cumpla ninguno de ellos. Si se cumple el primer predicado, la tupla de entrada se envía a la primera salida. Si no se cumple el primer predicado, pero se cumple el segundo, la tupla se envía a la segunda salida, y así sucesivamente. Si hay n + 1 salidas y n predicados, y una tupla no cumple ninguno de los predicados, la tupla se envía a la salida n + 1. Si hay n salidas y n predicados, y la tupla no cumple ningún predicado, la tupla se descarta, sin producir tupla de salida alguna.

La semántica asumida para los operadores con estado es tal como sigue. Los operadores tienen una ventana deslizante para cada flujo de entrada. La longitud de la ventana se puede expresar en tiempo o en número de tuplas, por ejemplo, las tuplas recibidas en la última hora o las últimas 1000 tuplas recibidas. Cuando llega una nueva tupla, esta se inserta en la ventana aplicando la semántica del operador. Como resultado, las tuplas más antiguas que excedan la longitud de la ventana son eliminadas. Por ejemplo, si la longitud de ventana es de 60 minutos y la diferencia entre las marcas de tiempo entre la tupla recién insertada y la primera tupla es de 61 minutos, la primera tupla se eliminada de la ventana. Lo mismo tiene lugar con la segunda tupla de la ventana y así sucesivamente.

El operador de agregación tiene una entrada y una salida. El operador de agregación tiene asociada una función de agregación fa, el conjunto de campos ca que se emplea para agrupar datos (condición de agrupamiento) y el avance o deslizamiento de ventana av. La función de agregación se aplica a las tuplas de la ventana deslizante. El resultado es el valor agregado de todas las tuplas en la ventana, por ejemplo, la temperatura promedio sobre tuplas que muestran la evolución de la temperatura con el tiempo. El conjunto de campos indicados en la condición de agrupamiento determina qué tuplas se agregan de forma conjunta. Por ejemplo, se puede agregar el número de llamadas por número de teléfono de llamante, en tuplas de registro de descripción de llamadas para llamadas telefónicas. Es decir, cada tupla de salida mostraría el número de llamadas realizadas por cada número de teléfono que haya realizado una llamada durante el período considerado por la ventana deslizante actual. Si se encuentra presente la condición de agrupamiento (es decir, agrupamiento por un campo dado o un conjunto de campos, cada valor diferente de ese campo o campos se encuentra en una ventana deslizante separada. Por ejemplo, si el número de teléfono de llamante se encuentra en la condición de agrupamiento, cada número de teléfono tendrá una ventana deslizante asociada que contendrá las tuplas que se corresponden con un teléfono llamante dado. El avance indica cuánto se desliza la ventana deslizante. Si esta es una ventana de tiempo, el avance indica durante cuanto tiempo se mueve la ventana y. si esta es una longitud de ventana, el avance indica cuántas tuplas se deslizan. Por ejemplo, en una ventana de tiempo de 60 minutos con un avance de 15 minutos, cuando se inserta una tupla de tal modo que la diferencia entre su marca de tiempo y la tupla más antigua en la ventana supera los 60 minutos, se genera una tupla de salida con el valor agregado durante los últimos 60 minutos. Se eliminan de la ventana todas las tuplas con una diferencia más grande que 60 - 15 = 45 minutos en comparación con la tupla que se acaba de insertar. Es decir, el operador de agregación producirá tuplas de salida con la misma periodicidad que indica el avance. Si la longitud de la ventana se expresa en número de tuplas, un ejemplo de avance podría ser el siguiente. Con una longitud de ventana de 100 tuplas y un avance de 25 tuplas, cuando llega la tupla 101ª, se genera una tupla con la agregación de las primeras 100 tuplas de la ventana; entonces, se inserta la nueva tupla en la ventana y se eliminan las primeras 25 tuplas (las 25 tuplas más antiguas).

El operador de reunión tiene dos entradas, cada una con su propia ventana deslizante y un predicado de reunión asociado que recibe una tupla de cada entrada. Si un par de tuplas cumplen el predicado, se genera una tupla de salida con la concatenación de esas dos tuplas. Cada vez que llega una nueva tupla de entrada, la tupla se empareja contra las tuplas almacenadas en la ventana deslizante de la otra entrada, para cada par que cumple el predicado con la tupla entrante, y se genera una tupla de salida. Cuando se inserta una nueva tupla en una ventana deslizante, se eliminan las tuplas de la otra ventana deslizante más allá de la longitud de la ventana. Para la paralelización transparente, se considera que el operador de reunión es un operador determinista. Este operador de reunión espera hasta que hay una tupla en cada entrada antes de insertar una tupla en las ventanas deslizantes. Cuando hay una tupla en cada entrada, este coge la que tiene la marca de tiempo más pequeña y procede como se ha indicado en lo que antecede. De esta forma, el operador de reunión es independiente de la intercalación de las tuplas en sus dos entradas y, de esta forma, se comporta de forma determinista con independencia de las intercalaciones reales.

El método asumido para propagar las marcas de tiempo cada vez que se genera una tupla de salida es tal como sigue. Para los operadores sin estado, la tupla de salida tiene la misma marca de tiempo que la tupla de entrada de la que proviene la misma. En los operadores con estado con un único flujo de entrada, tales como el operador de agregación, la marca de tiempo de una tupla de salida es la marca de tiempo más pequeña en la ventana de tiempo en la que se computó la tupla de salida. Para el operador de reunión, la marca de tiempo de cada tupla de salida se

corresponde con la marca de tiempo más baja entre el par de tuplas de entrada reunidas.

El motor distribuido en paralelo consiste en un conjunto de instancias de un operador de consultas distribuido, un compilador de consultas paralelo y un conjunto de operadores de paralelización y de elasticidad. Hay dos tipos de operadores de paralelización, el equilibrador de carga y el mezclador de entradas. Los operadores de elasticidad son los aprovisionadores y el gestor de nodos disponibles.

La entrada del compilador paralelo de consultas es una consulta continua, es decir, un grafo acíclico de operadores de consulta. El compilador divide la consulta de origen en subconsultas, consistiendo cada subconsulta en un subconjunto de operadores de la consulta de origen que están interconectados. El compilador paralelo permite diferentes estrategias de paralelización. Una estrategia consiste en que cada operador individual se vuelva una subconsulta. Otra posible estrategia consiste en tener una única subconsulta que contiene la consulta de origen completa. Una estrategia intermedia consiste en subdividir la consulta de origen en tantas subconsultas como operadores con estado, más un prefijo opcional de operadores sin estado que se encuentran al principio de la consulta de origen. Cada subconsulta tiene un operador con estado, seguido de la totalidad de los operadores sin estado hasta la siguiente subconsulta o subconsultas. Si la consulta comienza por algún operador sin estado, hay una subconsulta adicional con la totalidad de los operadores sin estado previos hasta el primer operador u operadores de consulta con estado. Esta estrategia de paralelización (partición en subconsultas con estado) minimiza el coste de red. Esta solo envía tuplas a través de la red solo cuando es necesario (para preservar la transparencia), es decir, justo junto antes de cada operador de consulta con estado. Sin pérdida de generalidad, se asumirá esta estrategia de paralelización en el resto de la descripción.

La consulta de origen se compila mediante el compilador paralelo, el resultado es un conjunto de subconsultas. Se asume la estrategia de partición en subconsultas con estado que se ha descrito en lo que antecede. Cada subconsulta se complementa del siguiente modo. Al principio de la subconsulta se introduce un mezclador de entradas. Al final de cada salida de subconsulta se introduce un equilibrador de carga. Cada subconsulta se despliega en un conjunto diferente de nodos. En cada nodo se ejecuta una instancia del motor de procesamiento de flujos. El despliegue de una subconsulta en un conjunto de nodos radica en el despliegue de esa subconsulta en la instancia del motor de procesamiento de flujos en cada nodo. Las subconsultas se conectan tal como sigue. Cada conexión entre una salida de la subconsulta de origen con una entrada de una subconsulta de destino en la consulta original da como resultado una conexión entre el equilibrador de carga de esa salida en cada instancia de la subconsulta de origen y el mezclador de entradas de cada instancia de la subconsulta de destino. La figura 5 muestra un ejemplo de la conexión de subconsultas paralelas, que se corresponde con la partición en subconsultas con estado que se muestra en la figura 3, para la consulta de origen en la figura 1.

Los equilibradores de carga son conscientes de la semántica. Es decir, distribuyen la carga entre los nodos de la siguiente subconsulta (o subconsultas) de tal modo que las tuplas que tengan que se hayan de combinar de forma conjunta en el mismo operador de agregación o de reunión, sean recibidas y, por lo tanto, procesadas por el mismo nodo. La distribución consciente de la semántica se puede implementar de cualquier forma, pero esta ha de cumplir este requisito, es decir, las tuplas que se han de combinar de forma conjunta en envían a la misma instancia de la subconsulta de destino. En el presente modo de realización preferido se adoptará la siguiente distribución consciente de la semántica. Un operador con estado de una subconsulta de destino emplea un campo o un conjunto de campos, C, para combinar tuplas, que se denomina clave de tupla. Para el operador de reunión, la clave consiste en los campos que se emplean en el predicado de reunión. Para el operador de agregación, la clave consiste en los campos en la condición de agrupamiento. De este modo, cada tupla de salida tiene una clave c. A esta clave c se le aplica una función hash, obteniendo el valor h = hash (c). Al valor h se le aplica la operación resto con una constante np, obteniendo un valor p = h mod np, que se denomina identificador de partición. El resultado de la aplicación del método anterior a una tupla es el id de partición. El número total de particiones es np, y la responsabilidad de procesar una partición con el identificador p corresponde a una única instancia de las subconsultas. Cada instancia es responsable de un subconjunto de particiones. Una única instancia es responsable del procesamiento de las tuplas de una partición dada.

Si los mezcladores de flujo no son transparentes, estos simplemente reenvían las tuplas que se reciben de las instancias de la subconsulta de origen, tan pronto como estas se reciben, a su instancia local de la subconsulta.

Para garantizar la paralelización transparente, se extienden los equilibradores de carga y los mezcladores de entradas. Los mezcladores de entradas esperan a recibir una tupla de cada instancia de la subconsulta de origen y, cuando esto ocurre, los mismos reenvían la tupla con la marca de tiempo más pequeña a su instancia local de su subconsulta. Los mezcladores de entradas se pueden bloquear con esta mezcla de flujos si no se reciben tuplas de uno de los equilibradores de carga. Los equilibradores de carga se extienden para producir tuplas espurias cuando los mismos no han producido tupla alguna durante un periodo de tiempo dado para evitar el bloqueo. Cada equilibrador de carga realiza, por lo tanto, un seguimiento de la marca de tiempo de la última tupla generada para cada instancia de la subconsulta (subconsultas) de destino. Cuando a una subconsulta de destino no se le ha enviado tupla alguna durante un periodo de tiempo máximo m, entonces se envía una tupla espuria con la marca de tiempo de la última tupla enviada por ese equilibrador de carga (a cualquiera de las instancias de la subconsulta de destino). La generación periódica de tuplas espurias evita el bloqueo del mezclador de entradas cuando un

equilibrador de carga no genera tuplas para una instancia de una subconsulta de destino y de tal modo que el mezclador de entradas pueda progresar.

La paralelización de subconsultas sin estado, es decir, subconsultas que consisten de forma exclusiva en operadores sin estado, requiere simplemente una distribución de tipo por orden cíclico de las diferentes particiones entre las instancias de la subconsulta. Es decir, se envían tuplas a cada una de las instancias de la subconsulta de destino hasta que se haya enviado una tupla a todas ellas, entonces se empieza de nuevo con la primera. Por ejemplo, en una subconsulta de destino con dos instancias, la primera tupla se enviaría a la primera instancia, la segunda tupla a la segunda instancia, la tercera tupla a la primera instancia, y así sucesivamente.

La paralelización de subconsultas con operadores con estado es más compleja debido a que la misma ha de ser consciente de la semántica de los operadores con estado de la subconsulta de destino. En este caso, se ha de dividir el espacio de claves de las tuplas de salida, de tal modo que a cada clave se le asigna un identificador de partición, empleando, por ejemplo, el método de aplicación de hash que se ha mencionado en lo que antecede. De este modo, dada una tupla de salida con una clave c, se le asigna un identificador de partición p. Por otro lado, a cada partición se le asigna una y solo una de la subconsulta que será responsable de procesar todas las tuplas con ese identificador de partición. Ese método se emplea para el operador de agregación. Las tuplas con la misma clave son recibidas por la misma instancia y, de esta forma, estas se pueden agregar localmente de forma conjunta. Este método de distribución también es válido para la reunión con un predicado de igualdad (*equi-join*, equi-reunión). En este caso, las tuplas con la misma clave, que son las únicas para las que se debería comprobar el predicado, son recibidas por la misma instancia de la subconsulta de destino y, de esta forma, puede reunir las tuplas con misma clave. Por ejemplo, para reunir las tuplas de llamadas telefónicas con el mismo número de teléfono llamante, el predicado de reunión requiere que el campo del teléfono llamante de las dos tuplas a comparar tenga el mismo valor. Debido a que las tuplas con el mismo número de teléfono llamante son recibidas por la misma instancia de la subconsulta de destino, estas se pueden reunir localmente. No obstante, para el caso general del operador de reunión (al igual que tiene lugar con el caso del operador de producto Cartesiano), cuando no hay un predicado de igualdad que haya de ser cumplido por las tuplas a reunir, se emplea un método de distribución diferente. Para simplificar la exposición, se asume que el número de instancias de la subconsulta con el operador de reunión es un cuadrado 1, 4, 9, 16, ... El número de instancias es $i = j^2$. Las instancias se numeran de 0 a i - 1. El operador de reunión tiene dos entradas que se denominarán izquierda y derecha. Los equilibradores de carga conectados a la entrada izquierda enviarán una tupla de salida a i instancias de la subconsulta de destino con el operador de reunión. El identificador de tupla de salida que se está considerando es p. Más en concreto, la tupla de salida se envía a i instancias numeradas d = p * j + o, en la que o toma valores de 0 a j - 1. Los equilibradores de carga conectados a la entrada derecha también enviarán la tupla de salida a i instancias de la subconsulta de destino numeradas con d = p + o * j, en la que o toma valores de 0 a j - 1. Esto garantiza que se generen todos los pares posibles de entre las tuplas con una distancia temporal más pequeña o igual que la longitud de la ventana de tiempo que está asociada con el operador de reunión.

A continuación se describe cómo extender el método de paralelización para obtener elasticidad, equilibrado de carga y tolerancia a fallos. Tal como se ha comentado en lo que antecede, las tres propiedades requieren reconfigurar el sistema y la reconfiguración necesita de un procedimiento de transferencia de estado. Para transferir la responsabilidad del procesamiento de una subconsulta que es responsable de una partición con el identificador p, de una instancia A a otra instancia B de la misma subconsulta, el estado del operador con estado de la subconsulta en relación con esa partición se ha de transferir de A a B.

El procedimiento de transferencia de estado tiene las siguientes etapas. Cuando se toma la decisión de reconfigurar el sistema, la reconfiguración implica la reasignación de particiones de datos de algunas instancias a otras. Esta reasignación colectiva se puede descomponer en reasignaciones de particiones individuales de una partición con el identificador p de una instancia A a otra instancia B. En primer lugar, se escoge una marca de tiempo, mt, en la que comenzará la transferencia de estado. Esta marca de tiempo, mtf, es más grande que la marca de tiempo más grande, mt, que esté activa en el conjunto de nodos de la subconsulta para la que se realiza la reconfiguración, de tal modo que mtf = mt + f, en la que f proporciona un margen suficiente para notificar a todas las instancias involucradas en la reconfiguración que han de iniciar la transferencia de estado con aquellas tuplas con una marca de tiempo igual o más grande que mtf. La marca de tiempo mtf se comunica a todas las instancias de la subconsulta (subconsultas) de origen que envía tuplas a la subconsulta reconfigurada (que se denominará destino), así como a las dos instancias de la subconsulta de destino reconfigurada, A y B. Los equilibradores de carga de las instancias de la subconsulta de origen almacenan la marca de tiempo mtf. Las únicas tuplas que se distribuyen de una forma diferente son aquellas que pertenecen a la partición p, el resto no varía su procesamiento. Cada equilibrador de carga redistribuye las tuplas que este recibe en orden creciente de marca de tiempo. Cada equilibrador de carga reenvía las tuplas que este recibe de la partición p con una marca de tiempo inferior o igual a mtf a la instancia A de la subconsulta de destino, la responsable inicial para la partición p. Cuando un equilibrador de carga recibe la primera tupla con una marca de tiempo más grande que mtf, este envía esa tupla a la instancia B, la nueva responsable de la partición p, y una tupla de fin de reconfiguración con el identificador de partición se envía justo antes de esa tupla. En el caso de un operador de reunión con un predicado sin igualdad, hay un conjunto de instancias A con la responsabilidad original de la partición p y un conjunto de instancias B con la responsabilidad de destino. El procedimiento se aplica de forma similar a la totalidad de los conjuntos de instancias.

Además de la reconfiguración de los equilibradores de carga, el estado que se corresponde con la partición p se transfiere de la instancia de origen A a la instancia de destino B. Para el operador de agregación, este estado consiste en el valor agregado para la partición p y la ventana de tiempo de la partición p. Para el caso del operador de reunión con igualdad en el predicado, este en las tuplas de las dos ventanas deslizantes que están asociadas a la partición p. En el caso del operador de reunión sin igualdad en el predicado, este consiste en las tuplas que están asociadas a la partición de datos transferida. La instancia B espera hasta que la transferencia de estado se ha completado y a recibir todas las tuplas de fin de reconfiguración por parte de todos los equilibradores de carga. En ese momento, la instancia B comienza a procesar las tuplas de la partición p.

Se describe en lo sucesivo el procedimiento que se emplea para extender el procedimiento de paralelización para conseguir elasticidad y distribución de carga. Para cada subconjunto de nodos que procesan una subconsulta, un nodo se vuelve el aprovisionador. Cada nodo del subconjunto de nodos monitoriza periódicamente la carga local en cada nodo (por medio de una métrica directa tal como un porcentaje de utilización de la CPU o una métrica indirecta tal como el número de tuplas puestas en cola pendientes de procesar en la instancia de la subconsulta). La información de monitorización se envía de forma periódica de cada nodo del subconjunto de nodos al aprovisionador. Este mensaje de información también incluye la marca de tiempo más grande procesada por la instancia. Por lo tanto, el aprovisionador conoce la carga relativa de todos los nodos que ejecutan una instancia de la subconsulta. El aprovisionador compara la carga relativa entre los nodos. Si el desequilibrio entre el nodo más cargado y el menos cargado supera un umbral de desequilibrio superior, se reconfigura la subconsulta al mover una o más particiones de datos del nodo más cargado al menos cargado. El procedimiento de reconfiguración para cada partición sigue el que se ha descrito en lo que antecede para la transferencia de estado.

El aprovisionador también calcula la carga promedio de los nodos y, si esta carga promedio supera otro umbral de utilización, esto quiere decir que el conjunto de nodos está cercano a la saturación y se aprovisiona un nodo libre. En primer lugar, una instancia del motor de procesamiento de consultas se despliega en ese nodo con una copia de la subconsulta. A continuación, el equilibrador de carga de la nueva instancia se conecta con los mezcladores de flujo a los que está conectada la salida o salidas de la subconsulta. Se hace lo mismo con la entrada o entradas del mezclador o mezcladores de entradas de la subconsulta y las salidas de los equilibradores de carga de la subconsulta a la que están conectadas sus entradas. A partir de ese momento, la instancia de la subconsulta empieza a informar acerca de su carga, que inicialmente es nula. Esto desencadena el procedimiento de equilibrado de carga que se ha descrito en lo que antecede.

El aprovisionador también comprueba si la carga global actual de la subconsulta se podría cumplir con un nodo menos sin superar el umbral de utilización superior de la carga promedio. En ese caso, todas las particiones del nodo menos cargado se redistribuyen de manera uniforme entre el resto de los nodos. Cada partición se reconfigura con el mismo procedimiento que se describe para el equilibrado de carga. Cuando se ha reconfigurado la totalidad de las particiones de datos, se desconectan las salidas de los equilibradores de carga que están conectados con las entradas del nodo a retirar del servicio y las salidas de los equilibradores de carga también se desconectan de las entradas de la siguiente subconsulta (subconsultas). Una vez que la instancia de la subconsulta del nodo a retirar del servicio se ha desconectado, ese nodo se retira del servicio y se devuelve al conjunto de nodos libres.

El motor distribuido en paralelo propuesto y su procedimiento de procesamiento paralelo de consultas se pueden desplegar en un sistema de computación en la nube. Más en concreto, este se puede desplegar en una infraestructura como un servicio responsable de la gestión de los nodos libres. En ese caso, el procedimiento de elasticidad se modifica tal como sigue. El aprovisionador de cada subconsulta delega la tarea de la gestión de nodos, consiguiendo y liberando nodos, en la infraestructura como servicio. Por lo tanto, cuando se va a aprovisionar un nuevo nodo el aprovisionador en cuestión solicita un nuevo nodo a la infraestructura como servicio. Cuando se libera un nodo, se le notifica a la infraestructura como servicio que el nodo está libre.

La invención es aplicable al sector industrial de los sistemas de información de procesamiento de flujos de datos y de procesamiento de eventos. En ambos tipos de sistemas se procesan flujos de datos mediante consultas continuas. Las soluciones actuales son o bien centralizadas o bien distribuidas, pero en ambos casos la capacidad de procesamiento de estos sistemas está limitada por la capacidad de procesamiento de un único nodo debido a que la totalidad del flujo de datos de entrada es procesado por un único nodo para procesar un operador de consulta dado, una subconsulta o una consulta completa. Esta limitación evita que los sistemas actuales escalen con respecto al volumen del flujo de datos de entrada.

REIVINDICACIONES

1. Un motor de procesamiento paralelo de flujos de datos que ejecuta consultas continuas sobre una pluralidad de nodos de procesamiento (1), en el que una consulta continua comprende una pluralidad de operadores interconectados, siendo seleccionable cada operador de la consulta entre un operador sin estado (M, F) y un operador con estado (A, J), que comprende:

a) unos medios para dividir una consulta continua en al menos una subconsulta, en el que cada una de dicha al menos una subconsulta consiste en un operador con estado (A, J) seguido por al menos un operador sin estado (M, F), excepto por una subconsulta inicial que solo contiene operadores sin estado (M, F),

b) unos medios para asociar cada al menos una subconsulta con al menos dos nodos (1); en el que los nodos (1) están configurados para ejecutar la subconsulta; en el que una subconsulta ejecutada genera al menos una tupla;

c) unos medios para etiquetar las tuplas generadas con una etiqueta de orden que comprende una marca de tiempo que indica un orden relativo entre las tuplas y que se emplea para ordenar las tuplas a una subconsulta de destino (3) en el mezclador de entradas (IM)

caracterizado por que este comprende adicionalmente

d) unos medios para enviar y recibir tuplas entre subconsultas interconectadas, una subconsulta de origen (2) y una subconsulta de destino (3) que comprenden operadores interconectados entre las mismas; en el que los medios para enviar y recibir tuplas comprenden:

- para cada nodo (1) que ejecuta una subconsulta de origen (2), un equilibrador de carga (LB) que está configurado para enviar una pluralidad de tuplas generadas a los nodos (1) que ejecutan una subconsulta de destino (3);

- para cada nodo (1) que ejecuta una subconsulta de destino, un mezclador de entradas (IM) que está configurado para recibir una pluralidad de tuplas generadas que son enviadas por cada equilibrador de carga (LB) de la subconsulta de origen;

en el que el equilibrador de carga (LB) está configurado para determinar el nodo (1) al que se envía una tupla en función de al menos un campo comprendido en la tupla, campo que se denomina campo de clave;

y en el que para cada subconsulta que comprende un operador sin estado (M, F) o un operador con estado (A, J):

- se introduce un mezclador de entradas (IM) al inicio de cada operador (A, M) y un equilibrador de carga (LB) al final de cada operador (A, M), en donde se ejecuta la subconsulta, y

en el que cada equilibrador de carga (LB) está conectado con cada mezclador de entradas (IM) respectivamente para distribuir las tuplas de salida de la subconsulta de origen (2) a la subconsulta de destino (3), en el que cada equilibrador de carga está configurado para:

- enviar tuplas con un mismo campo de clave a un mismo nodo (1) que ejecuta la subconsulta de destino (3), cuando el operador es un operador con estado (A, J), o

- enviar tuplas a la subconsulta de destino de una forma por orden cíclico, cuando el operador es un operador sin estado (M, F).

2. El motor de acuerdo con la reivindicación 1, en el que los medios para dividir consultas emplean el método para dividir la consulta de origen en tantas subconsultas como operadores están incluidos en la consulta original.

3. El motor de acuerdo con cualquiera de las reivindicaciones 1 a 2, en el que el envío de tuplas que se reciben con la misma clave al nodo (1) que ejecuta una subconsulta de destino (3) comprende:

a) aplicar a cada tupla con clave c una función hash, obteniendo un valor h = hash;

b) obtener el identificador de partición p de la tupla, realizando la operación resto por una constante np, obteniendo un valor p = h mod np;

c) asignar cada partición con el identificador p a un nodo (1) que ejecuta una subconsulta de destino (3).

4. El motor de acuerdo con la reivindicación 1, en el que:

a) cada mezclador de entradas (IM) reenvía tuplas cuando un mezclador de entradas (IM) recibe las tuplas de cualquiera de las subconsultas de origen a la subconsulta de destino (3) con la que está conectada, o;

b) cada mezclador de entradas (IM) espera a recibir una tupla de cada uno de los flujos de entrada del mezclador de entradas (IM) antes de reenviar la tupla con la marca de tiempo más pequeña a la subconsulta de destino (3) con la que está conectada,

c) un equilibrador de carga (LB) almacena la marca de tiempo de la última tupla generada para cada subconsulta de destino (3) y, si después de un periodo de tiempo previamente fijado máximo m, no se ha enviado una tupla a la subconsulta de destino (3), el equilibrador de carga (LB) envía una tupla espuria con la misma marca de tiempo que tenía la última tupla enviada por el equilibrador de carga (LB).

5. El motor de acuerdo con cualquiera de las reivindicaciones 2 a 4, en el que este reconfigura adicionalmente el procesamiento de tuplas que se corresponden con una partición p, al transferir el procesamiento de dichas tuplas de un nodo de origen (1) a un nodo de destino (1), que comprende:

a) unos medios para obtener la marca de tiempo activa más grande en el sistema, mt;

b) unos medios para establecer una marca de tiempo futura mtf, mediante la adición de un desplazamiento temporal f a la marca de tiempo activa más grande en el sistema, mt, mtf = mt + f;
c) unos medios para enviar, durante el margen de tiempo establecido f, una orden de reconfiguración que incluye la marca de tiempo que inicia la reconfiguración, mtf, a todas las subconsultas de origen que están involucradas en la reconfiguración;
d) unos medios para almacenar la marca de tiempo mtf en todos los equilibradores de carga (LB) de la subconsulta de origen (2);
e) el envío de las tuplas que se corresponden con la partición reconfigurada p con una marca de tiempo más pequeña o igual que mtf por los equilibradores de carga de la subconsulta de origen al nodo (1) que ejecuta la subconsulta de destino responsable de la partición p antes de que se iniciara la reconfiguración;
f) unos medios para el envío de una tupla por los equilibradores de carga (LB) de la subconsulta de origen (2) que indica el fin de la reconfiguración a los nodos (1) que ejecutan la subconsulta de destino (3) que está involucrada en la transferencia de estado antes de enviar la primera tupla de la partición p más grande que mtf;
g) unos medios para el envío de las tuplas que pertenecen a la partición reconfigurada p con una marca de tiempo más grande que mtf por los equilibradores de carga de la subconsulta de origen al nodo (1) que ejecuta la subconsulta de destino (3) responsable de la partición p después de la reconfiguración;
h) unos medios para transferir el estado de la partición p al nodo de destino (1) después de recibir la tupla que indica el fin de la reconfiguración de todos los equilibradores de carga (LB) de la subconsulta de origen (2);
i) unos medios para aplicar en el nodo de destino (1) el estado de la partición p después de recibir la transferencia de estado de la partición p.

6. El motor de acuerdo con la reivindicación 5, en el que la configuración de uno de los nodos (1) en los que se ejecuta una subconsulta para equilibrar la carga comprende:
a) unos medios para recibir de forma periódica, de la totalidad de los nodos (1) en los que se está ejecutando la subconsulta, datos acerca de la utilización de CPU y de memoria en cada nodo (1), y la marca de tiempo más grande de entre las tuplas procesadas en cada nodo (1),
b) unos medios para comparar los datos de utilización entre los nodos (1) que ejecutan la subconsulta;
c) unos medios para enviar una orden de reconfiguración a los nodos (1) involucrados de la subconsulta de origen (2) y la subconsulta de destino (3), si, para una subconsulta, la comparación entre la utilización supera un primer umbral de utilización superior dado, el nodo (1) reconfigura al menos una partición de datos,
d) unos medios para añadir el nodo (1) seleccionado al conjunto de nodos (1) que ejecuta la subconsulta de tal modo que el nodo (1) seleccionado recibirá carga de otros nodos (1) de forma automática, si la carga promedio de los nodos (1) de una subconsulta supera un segundo umbral de desequilibrio superior dado, el nodo (1) selecciona un nodo (1) de entre el conjunto de nodos libres (1),
e) unos medios para enviar una orden de reconfiguración al nodo (1) seleccionado de tal modo que la totalidad de las particiones de los nodos (1) seleccionados se distribuirán entre el resto de los nodos (1) que ejecutan la subconsulta, si la carga promedio de los nodos (1) de una subconsulta se puede cumplir con un conjunto más pequeño de nodos (1) sin superar el segundo umbral de utilización superior, el nodo (1) selecciona un nodo (1) de entre el conjunto de nodos (1) en los que se está ejecutando la subconsulta.

7. Un método de procesamiento paralelo de flujos de datos que ejecuta consultas continuas sobre una pluralidad de nodos de procesamiento (1), en el que una consulta continua comprende una pluralidad de operadores interconectados, siendo seleccionable cada operador de la consulta entre un operador sin estado (M, F) y un operador con estado (A, J), que comprende:
a) dividir una consulta continua en al menos una subconsulta, en el que dicha subconsulta consiste en un operador con estado (A, J) seguido por al menos un operador sin estado (M, F), excepto por una subconsulta inicial que solo contiene operadores sin estado (M, F),
b) asociar cada al menos una subconsulta con al menos dos nodos (1); en el que los nodos (1) están configurados para ejecutar la subconsulta; en el que una subconsulta ejecutada genera al menos una tupla;
c) etiquetar las tuplas generadas con una etiqueta de orden que comprende una marca de tiempo que indica un orden relativo entre las tuplas y que se emplea para ordenar las tuplas a una subconsulta de destino (3) en el mezclador de entradas (IM),

caracterizado por que este comprende adicionalmente
d) enviar y recibir tuplas entre subconsultas interconectadas, una subconsulta de origen (2) y una subconsulta de destino (3) que comprenden operadores interconectados entre las mismas; en el que los medios para enviar y recibir tuplas comprenden:
- para cada nodo (1) que ejecuta una subconsulta de origen (2), enviar una pluralidad de tuplas generadas a los nodos (1) que ejecutan una subconsulta de destino (3);
- para cada nodo (1) que ejecuta una subconsulta de destino, recibir una pluralidad de tuplas generadas que son enviadas por cada nodo (1) que ejecuta la subconsulta de origen (2);
en el que el equilibrador de carga (LB) está configurado para determinar el nodo (1) al que se envía una tupla en función de al menos un campo comprendido en la tupla, campo que se denomina campo de clave;

en el que para cada subconsulta que comprende un operador sin estado (M, F) o un operador con estado (A, J):

- se introduce un mezclador de entradas (IM) al inicio de cada operador (A, M) y un equilibrador de carga (LB) al final de cada operador (A, M), en donde se ejecuta la subconsulta, y
en el que cada equilibrador de carga (LB) está conectado con cada mezclador de entradas (IM) respectivamente para distribuir las tuplas de salida de la subconsulta de origen (2) a la subconsulta de destino (3), en el que cada equilibrador de carga está configurado para:
- enviar tuplas con un mismo campo de clave a un mismo nodo (1) que ejecuta la subconsulta de destino (3), cuando el operador es un operador con estado (A, F), o;
- enviar tuplas a la subconsulta de destino de una forma por orden cíclico, cuando el operador es un operador sin estado (M, F).

8. El método de acuerdo con la reivindicación 7, en el que los medios para dividir consultas dividen la consulta de origen (2) en tantas subconsultas como operadores están incluidos en la consulta original.

9. El método de acuerdo con cualquiera de las reivindicaciones 7 a 8, en el que el envío de tuplas que se reciben con la misma clave al nodo (1) que ejecuta una subconsulta de destino (3) comprende:
a) aplicar a cada tupla con clave c una función hash, obteniendo un valor h = hash;
b) obtener el identificador de partición p de la tupla, realizando la operación resto por una constante np, obteniendo un valor p = h mod np;
c) asignar cada partición con el identificador p a un nodo (1) que ejecuta una subconsulta de destino (3).

10. El método de acuerdo con cualquiera de las reivindicaciones 7 a 9, en el que:
a) cada mezclador de entradas (IM) reenvía tuplas cuando un mezclador de entradas (IM) recibe las tuplas de cualquiera de las subconsultas de origen (2) a la subconsulta de destino (3) con la que está conectada,
b) cada mezclador de entradas (IM) espera a recibir una tupla de cada uno de los flujos de entrada del mezclador de entradas (IM) antes de reenviar la tupla con la marca de tiempo más pequeña a la subconsulta de destino (3) con la que está conectada,
c) un equilibrador de carga (LB) almacena la marca de tiempo de la última tupla generada para cada subconsulta de destino (3) y, si después de un periodo de tiempo previamente fijado máximo m, no se ha enviado una tupla a la subconsulta de destino (3), el equilibrador de carga (LB) envía una tupla espuria con la misma marca de tiempo que tenía la última tupla enviada por el equilibrador de carga (LB).

11. El método de acuerdo con cualquiera de las reivindicaciones 8 a 10, en el que este reconfigura adicionalmente el procesamiento de tuplas que se corresponden con una partición p, al transferir el procesamiento de dichas tuplas de un nodo de origen (1) a un nodo de destino (1), que comprende:
a) obtener la marca de tiempo activa más grande en el sistema, mt;
b) establecer una marca de tiempo futura mtf, mediante la adición de un desplazamiento temporal f a la marca de tiempo activa más grande en el sistema, mt, mtf = mt + f;
c) enviar, durante el margen de tiempo establecido f, una orden de reconfiguración que incluye
la marca de tiempo que inicia la reconfiguración, mtf, a todas las subconsultas de origen que están involucradas en la reconfiguración;
d) almacenar la marca de tiempo mtf en todos los equilibradores de carga (LB) de la subconsulta de origen (2);
e) el envío de las tuplas que se corresponden con la partición reconfigurada p con una marca de tiempo más pequeña o igual que mtf por los equilibradores de carga de la subconsulta de origen al nodo (1) que ejecuta la subconsulta de destino (3) responsable de la partición p antes de que se iniciara la reconfiguración;
f) el envío de una tupla por los equilibradores de carga (LB) de la subconsulta de origen (2) que indica el fin de la reconfiguración a los nodos (1) que ejecutan la subconsulta de destino (3) que está involucrada en la transferencia de estado antes de enviar la primera tupla de la partición p más grande que mtf;
g) el envío de las tuplas que pertenecen a la partición reconfigurada p con una marca de tiempo más grande que mtf por los equilibradores de carga de la subconsulta de origen al nodo (1) que ejecuta la subconsulta de destino (3) responsable de la partición p después de la reconfiguración;
h) transferir el estado de la partición p al nodo de destino (1) después de recibir la tupla que indica el fin de la reconfiguración de todos los equilibradores de carga (LB) de la subconsulta de origen;
i) aplicar en el nodo de destino (1) el estado de la partición p después de recibir la transferencia de estado de la partición p.

12. El método de acuerdo con la reivindicación 11, en el que la configuración de uno de los nodos (1) en los que se ejecuta una subconsulta para equilibrar la carga comprende:
a) recibir de forma periódica, de la totalidad de los nodos (1) en los que se está ejecutando la subconsulta, datos acerca de la utilización de CPU y de memoria en cada nodo (1), y la marca de tiempo más grande de entre las tuplas procesadas en cada nodo (1);
b) comparar los datos de utilización entre los nodos (1) que ejecutan la subconsulta;
c) si, para una subconsulta, la comparación entre la utilización supera un primer umbral de utilización superior dado, el nodo (1) reconfigura al menos una partición de datos, enviando una orden de reconfiguración a los nodos (1) involucrados de la subconsulta de origen y de destino;
d) si la carga promedio de los nodos (1) de una subconsulta supera un segundo umbral de desequilibrio superior dado, el nodo (1) selecciona un nodo (1) de entre el conjunto de nodos libres (1), añade el nodo (1) seleccionado

al conjunto de nodos (1) que ejecuta la subconsulta de tal modo que el nodo (1) seleccionado recibirá carga de otros nodos (1) de forma automática;

e) si la carga promedio de los nodos (1) de una subconsulta se puede cumplir con un conjunto más pequeño de nodos (1) sin superar el segundo umbral de utilización superior, el nodo (1) selecciona un nodo (1) de entre el conjunto de nodos (1) en los que se está ejecutando la subconsulta, y envía una orden de reconfiguración al nodo (1) seleccionado de tal modo que la totalidad de las particiones de los nodos (1) seleccionados se distribuirán entre el resto de los nodos (1) que ejecutan la subconsulta.

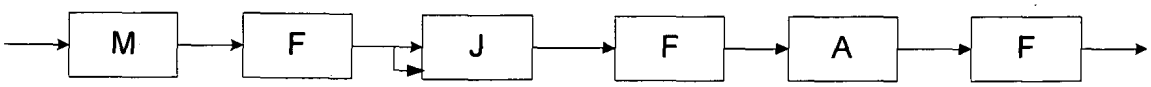

Figura 1

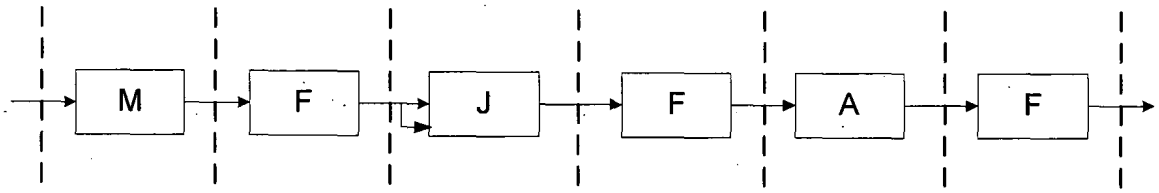

Figura 2

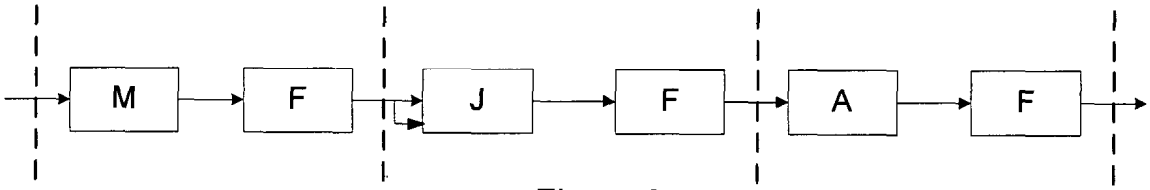

Figura 3

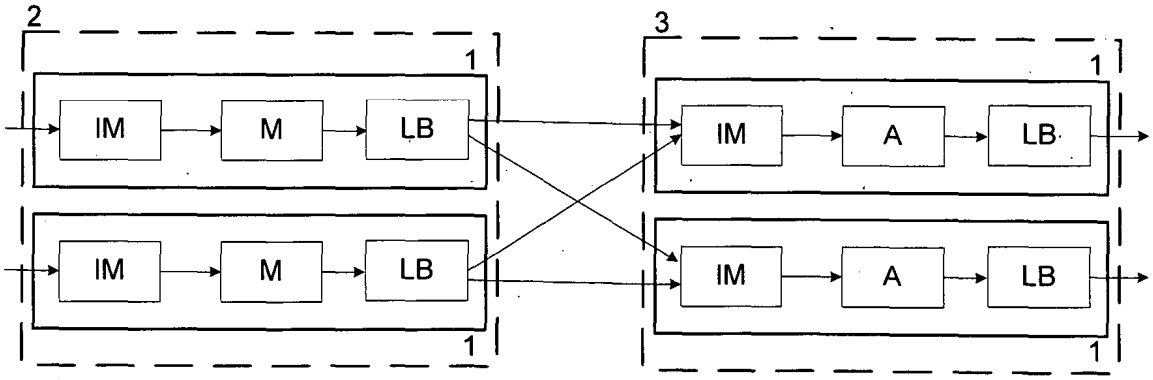

Figura 4

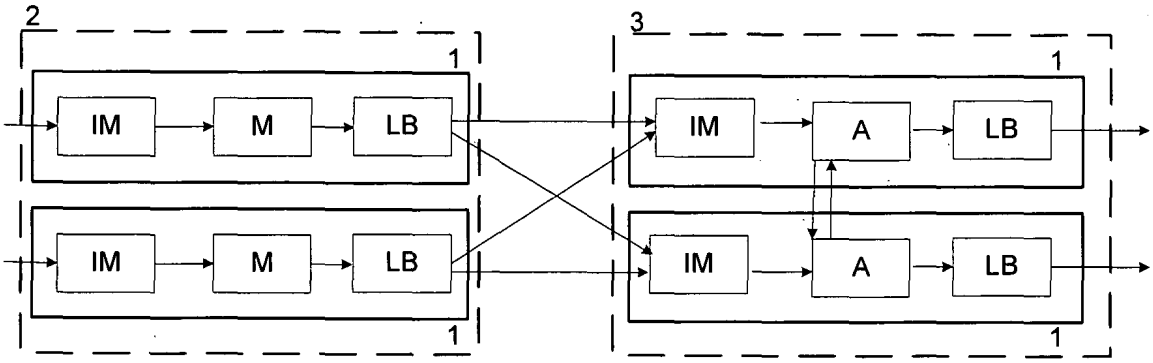

Figura 5